UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Agente conversacional para el apoyo al Centro de Psicología Aplicada

Junior Iván Soler Saba
Tutor: Pablo Alfonso Haya Coll
Ponente: German Montoro Manrique

Junio 2019

Agente conversacional para el apoyo al Centro de Psicología Aplicada

AUTOR: Junior Soler Saba
TUTOR: Pablo Alfonso Haya Coll

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2019

Resumen (castellano)

Este Trabajo Fin de Grado (TFG) se desarrolla la creación y soporte de una agente conversacional que permita la recogida de datos de potenciales pacientes del Centro de Psicología Aplicada (CPA) de la UAM.

Comenzaremos introduciendo el problema y la motivación detrás de este. A continuación, pasaremos a detallar algunos conceptos y tecnologías implicadas en el desarrollo de aplicaciones web y desarrollo de agentes conversacionales, prestando atención a funcionalidad que ofrecen dichas tecnologías. Viendo también el porqué de usar dichas tecnologías.

Una vez explicados dichos conceptos se procederán a analizar los distintos requisitos que deberá cumplir la aplicación y los distintos roles que tendrán los usuarios. Aquí también veremos algunos ejemplos de cómo se llegarían a realizar algunas de las funcionalidades especificadas.

Después veremos de manera más breve la arquitectura propuesta, de cada módulo de los que está compuesta esta y daremos algunos ejemplos de tecnologías que podrían proporcionarnos la funcionalidad requerida.

Posteriormente veremos de manera más detallada la arquitectura de la aplicación, el flujo de los datos y las tecnologías implicados en cada parte. Y ya sabiendo la arquitectura de esta mostraremos el flujo de navegación de la aplicación, las distintas partes de esta y cómo se comportará la misma en distintas situaciones.

Finalmente habiendo hecho un análisis de los resultados y las conclusiones obtenidas haremos un repaso de los objetivos y del proceso llevado a cabo para la construcción de la aplicación.

Por lo que en esta memoria se recogerán todos los aspectos del desarrollo de la aplicación para dar soporte al agente conversacional.

Palabras clave (castellano)

Chatbot, aplicación web, página web

Abstract (English)

This Bachelor Thesis develops the creation and support of a conversational agent that allows the collection of data from potential patients of the Center for Applied Psychology (CPA) of the UAM.

We will begin by introducing the problem and the motivation behind it. Then we will go on to detail some concepts and technologies involved in the development of web applications and development of conversational agents, paying attention to functionality offered by these technologies. Seeing also the reason for using these technologies.

Once explained these concepts will proceed to analyze the different requirements that the application must meet and the different roles that users will have. Here we will also see some examples of how some of the specified functionalities could be achieved.

Later we will see more briefly the proposed architecture, of each module of which it is composed, and we will give some examples of technologies that could provide us with the required functionality.

Later we will see in a more detailed way the architecture of the application, the flow of data and the technologies involved in each part. And knowing the architecture of this will show the navigation flow of the application, the different parts of it and how it will behave in different situations.

Finally, having made an analysis of the results and conclusions obtained we will review the objectives and the process carried out for the construction of the application.

So, in this report all aspects of the development of the application will be collected to support the conversational agent.

Keywords (inglés)

Chatbot, web application, web page

Agradecimientos

Me gustaría agradecer en primer lugar a mi madre, que es la que me apoya desde siempre y ha estado a mi lado desde el día uno hasta hoy.

También me gustaría agradecer tanto a Pablo, mi tutor, como a los psicólogos del CPA y a Helena, que sin su involucración este trabajo no habría sido posible.

Por último, también agradecer a amigos y compañeros por el apoyo brindado tanto ahora como en los años anteriores.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1 Introducción

1.1 Motivación

Esta memoria de TFG hace referencia al diseño y desarrollo del Chatbot LISA, una aplicación web con el objetivo de contener un chatbot que permita realizar una entrevista preliminar que permita recoger datos sobre potenciales pacientes del Centro de Psicología Aplicada (CPA) de la UAM.

Tanto las aplicaciones web como los chatbots o agentes conversacionales están muy presentes en nuestro día a día, en nuestros móviles o en algunas páginas web como puede ser AMAZON. El aprovechamiento de esta tecnología para la recogida de datos, en este caso, de posibles pacientes podrá ayudar mucho a los psicólogos, teniendo un pre-informe con el estado del paciente y algunos detalles más de los síntomas que padece este antes de que este siquiera haya tenido la primera consulta.

Para el desarrollo de este TFG se utiliza una tecnología de IBM, en concreto, Watson Assistant, tecnología proporcionada Bluemix, servicio en la nube de IBM. La lógica de la funcionalidad de este agente conversacional ha sido proporcionada por el laboratorio de lingüísticas informática de la UAM.

1.2 Objetivos

El principal propósito al realizar este TFG es la recogida de datos de potenciales pacientes del CPA, para lograr este propósito vemos que hay varios puntos:

- **Crear y diseñar una interfaz intuitiva y responsiva para interactuar con el agente conversacional** proporcionado por el departamento de lingüística informática.

- **Crear y diseñar de una interfaz de consulta de conversaciones y datos sobre los diferentes pacientes**. Además de crear toda la lógica de almacenamiento de estas conversaciones.

- **Crear un *"orquestador",*** componente que hará de puente entre la aplicación web y el agente conversacional.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1: Introducción**

  Introducción del trabajo. En este capítulo hablamos sobre la motivación del proyecto y los objetivos a llevar a cabo.

- **Capítulo 2: Estado del arte**

  En este capítulo veremos las diferentes tecnologías que envuelven al proyecto de manera teórica.

- **Capítulo 3: Análisis**

  En este apartado se incluye el análisis de los requisitos, tanto de los requisitos funcionales y como de los requisitos no funcionales, el alcance del proyecto. Se realiza un estudio de la funcionalidad inicial del proyecto con la finalidad de establecer las decisiones a tomar para el desarrollo de la aplicación.

- **Capítulo 4: Diseño**

  En esta sección hablaremos sobre la arquitectura de la aplicación y los diferentes sistemas involucrados en ella.

- **Capítulo 5: Desarrollo**

  Aquí veremos en detalle las diferentes tecnologías involucradas en cada parte de la arquitectura de la aplicación. Haciendo mención en detalle de algunos aspectos más críticos de la aplicación.

- **Capítulo 6: Integración, pruebas y resultados**

  Se exponen las diferentes pruebas realizadas durante el desarrollo del proyecto para asegurar la eficiencia, funcionamiento y resultados obtenidos del proyecto.

- **Capítulo 7: Conclusiones, discusión y trabajo futuro**

  Finalmente, en este capítulo se tratarán las conclusiones obtenidas y se expondrán algunas de las ideas acerca del trabajo futuro a realizar para la mejora de la aplicación.

2 Estado del arte

A lo largo de este capítulo trataremos de una manera más teórica los conceptos técnicos que envuelven al proyecto, también se mencionan a lo largo del capítulo las tecnologías utilizadas para el desarrollo de este.

2.1 Definiciones y conceptos básicos

A continuación, se detallan algunos conceptos generales relacionados con el ámbito de este documento:

- **Definición 2.1.1.** - *Chatbot* o bot de charla es aquel software que permite simular una conversación dando respuestas automáticas a entradas de realizadas por el usuario [1].
- **Definición 2.1.2.** - *JavaScript Object Notation* (JSON) es un formato de intercambio de datos ligeros. Está basado en un subconjunto del lenguaje de programación JavaScript [2]
- **Definición 2.1.3.** - *Application Programming Interface* (API) o Interfaces de programación de aplicaciones son un conjunto de comandos, funciones y protocolos informáticos que especifican cómo se comunican o interactúan un módulo de software con otro [3].
- **Definición 2.1.4.** - API REST, arquitectura que provee estándares entre los sistemas informáticos de la web, por lo cual se facilita la comunicación entre los distintos sistemas [4].
- **Definición 2.1.5.** - *Markup Language* (ML) o Lenguaje de marcado es un lenguaje que utiliza etiquetas para definir los diferentes elementos dentro de un documento [5].
- **Definición 2.1.6.** - *Extensible Markup Language* (XML) o Lenguaje de marcado extensible es un lenguaje de propósito general. Esto quiere decir que, en contrario a otros ML, en XML se deben definir las etiquetas a utilizar [6].
- **Definición 2.1.7.** - *Framework* o entorno de trabajo que provee las herramientas para el desarrollo de páginas web [7].
- **Definición 2.1.8.** - Lenguaje de programación interpretado es aquel lenguaje cuyo código se traduce a medida que se va siendo necesario, normalmente instrucción por instrucción [8].

2.2 Watson Assistant

Watson Assistant Conversation o *Watson Assistant* es una tecnología conversacional que nos da la posibilidad de crear chatbots. Este está integrado en la plataforma de computación en la nube de IBM conocida como Bluemix, esta plataforma permite utilizar otros servicios y tecnologías aparte de Watson Assistant [9].

Watson Assistant permite la creación de chatbots tanto simples como complejos atendiendo a las necesidades del desarrollador. La creación de los chatbots con él se facilita gracias a la interfaz gráfica que posee este. Watson Assistant permite la integración de otras tecnologías para dar una mayor complejidad al propio chatbot, como son servicios de traducción, conversión voz-texto o texto-voz... [10].

A continuación, vamos a ver algunos de los conceptos clave para el desarrollo del chatbot con Watson Assistant [10]:

- *"Intents"* (Intenciones): es el propósito u objetivo de los usuarios al escribir una entrada. Watson Assistant utiliza técnicas de machine learning y procesamiento de lenguaje natural para comprender las intenciones.
- *"Entities"* (Entidades): que sirven para clarificar y contextualizar de una manera más profunda las entradas de los usuarios.
- *"Dialogue Flow"* (Flujo de diálogo): es donde se define el flujo de las diferentes combinaciones de preguntas y respuestas según las diferentes entradas, intenciones, de los usuarios y el contexto proporcionado por la entidad.

2.2.1 API de Watson Assistant v1

Se puede utilizar las API REST de Watson Assistant y las herramientas de desarrollo de software correspondientes para la implementación de aplicaciones que deseen intercambiar información con Watson Assistant. Actualmente existen dos versiones de la API de Watson Assistant, v1 y v2, cada una con un conjunto de funciones diferente. En este caso nos centraremos en la versión v1.

Esta API nos proporciona la siguiente para el intercambio de información:

- *Authentication:* esta funcionalidad nos permitirá conectarnos y poder realizar peticiones a Watson Assistant. En la *figura 2-1*, podemos ver un ejemplo de conexión básica utilizando Python.

```python
from ibm_watson import AssistantV1

assistant = AssistantV1(
    version='{version}',
    username='{username}',
    password='{password}',
    url='{url}'
)
```

Figura 2-1: Ejemplo de *Authentication* Python

- *Mensajes:* con esto la API nos permite enviar las entradas de los diferentes usuarios a Watson Assistant. Al realizar esta petición Watson nos responderá con un objeto JSON con la respuesta a la entrada del usuario y distintas variables que son necesarias para la interacción con el chatbot. En la *figura 2-2* podemos ver un ejemplo de envío de mensaje y obtención de respuesta utilizando Python.

```python
import json
import ibm_watson

assistant = ibm_watson.AssistantV1(
    version='2019-02-28',
    iam_apikey='{apikey}',
    url='{url}'
)

response = assistant.message(
    workspace_id='{workspace_id}',
    input={
        'text': 'Hello'
    }
).get_result()

print(json.dumps(response, indent=2))
```

Figura 2-2: Ejemplo de petición de respuesta Python

2.3 Tecnologías para el desarrollo web

2.3.1 Python

Python es un lenguaje de programación interpretado creado a principios de los noventa por Guido Van Rossum. Este lenguaje es multiparadigma, ya que permite tanto la programación orientada objetos y programación imperativa, como, en menor grado, la programación funcional. Debido a que Python es un lenguaje de programación interpretado los errores aparecen en tiempo de ejecución. Las variables usadas en este lenguaje son de tipado fuerte, es decir, las variables de un tipo concreto no pueden usarse como si fueran de otro sin una conversión previa. También, estas variables son de tipado dinámico, puesto que una misma variable puede tener valores de distintos tipos [11] [12].

Una de las principales características fundamentales de Python en su sintaxis, que permite que la legibilidad y transparencia del código sean mucho mayores. Además de lo anteriormente mencionado, Python es multiplataforma, lo cual proporciona una gran ventaja a la hora de desarrollar nuestro código, independientemente del sistema en el que nos encontremos. Además, es muy fácil de extender, permitiendo la creación de forma simple de nuevos módulos en C o C++ [13].

2.3.2 Django

Django es un framework de desarrollo web escrito en Python, multiplataforma y de código abierto que anima al desarrollo rápido y al diseño pragmático y limpio. El objetivo de este framework es el desarrollo sencillo de sitios web complejos, haciendo hincapié en el re-uso, la conectividad y extensibilidad de sus componentes [14].

Este framework respeta el patrón de diseño conocido como Model-Template-View (MTV), a continuación, veremos la funcionalidad de cada uno de estos elementos [15], [16]:

- "Model" (Modelo): se encarga del intercambio de información y acceso a la base de datos. En esta capa se definen las características que poseerán los datos, también se define el cómo se accederá a estos, cómo se validarán, sus comportamientos y las distintas relaciones entre los diferentes datos.
- "Template" (Plantilla): parte encargada de la presentación. Esta capa contiene las decisiones relacionadas con la presentación, es decir, cómo son mostrados algunos elementos sobre una página web u otro tipo de documento.
- "View" (Vista): módulo de peticiones. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla correspondiente, es decir, actúa de puente entre los modelos y las plantillas.

Además del patrón de diseño MTV, Django posee una herramienta conocida como URLConf, que es un mapeo de las URLs que permite controlar el despliegue de las vistas. [16].

En la *figura 2-3* podemos observar el flujo que seguirá una petición en Django. El usuario realizará una petición desde un explorador de páginas web. Una vez dicha petición llegué a Django, el URLConf se encarga de leer la URL de la petición. Con esta URL busca la vista apropiada, que, a su vez, haciendo uso del modelo, obtiene la información necesaria de la base de datos y devuelve el resultado formateado según se exprese en la plantilla a utilizar [15].

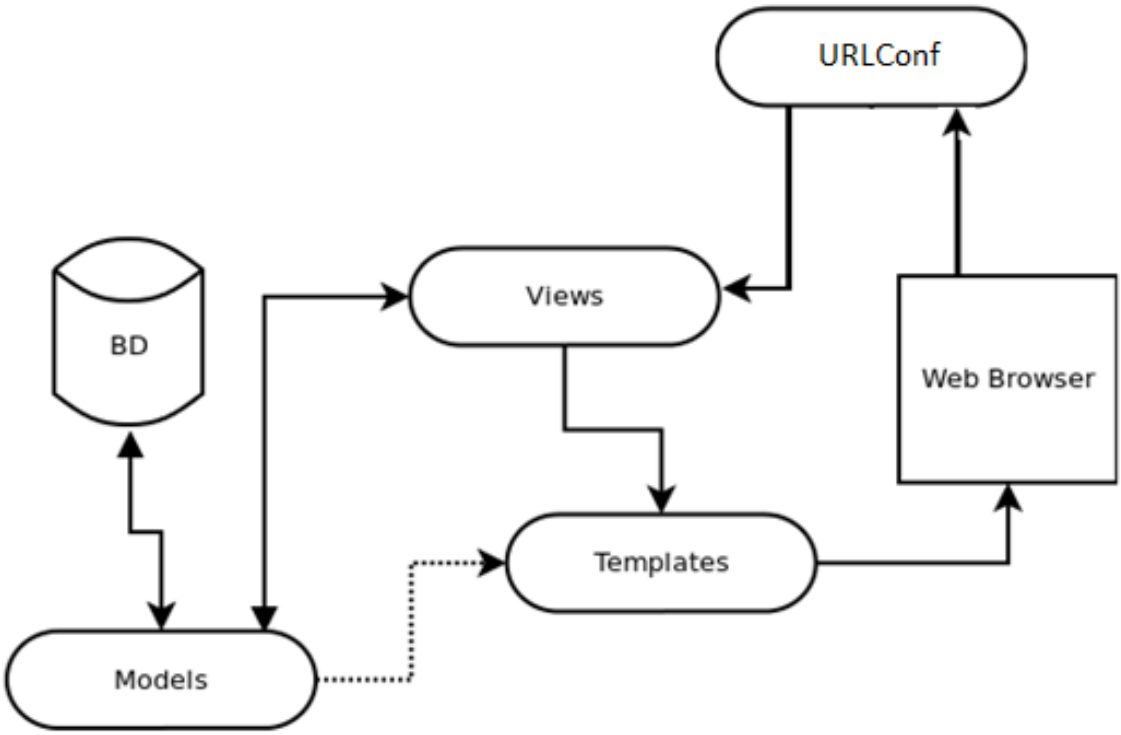

Figura 2-3: Flujo de peticiones en Django.

2.3.3 HTML5

El *HyperText Markup Language* (HTML) o Lenguaje de Marcación de Hipertexto es el lenguaje diseñado en 1989 para la creación de páginas web. Este lenguaje ha ido evolucionando con el tiempo de acuerdo a diversas necesidades, debido a esto, aparecieron nuevas versiones (HTML l, HTML 3.2, HTML 4, HTML 4.01) para mejorar dicho lenguaje de acuerdo a estas. En el año 2000, apareció xhtml (*eXtensible HyperText Markup Language*) como reformulación del HTML con sintaxis XML que también fue evolucionando, apareciendo nuevas versiones de este (xhtml 1, xhtml 2) debido a las diferencias entre diseñadores y organizaciones, hasta llegar a la versión HTML5 en el 2014, que mejoró y estandarizó muchos de los aspectos de la sintaxis de lenguaje [17].

Html5 incorpora nuevas etiquetas para la inserción de contenido multimedia (audio, vídeo y gráficos vectoriales) sin la necesidad de utilizar un plugin externo como Flash [17]. También incluye nuevas etiquetas para definir la estructura principal de la página web (footer, header...). Debido al uso de estas nuevas etiquetas y otras incluidas en HTML5 se reducen los tiempos de carga de las páginas web y se mejora facilita la creación de estas [18].

2.3.4 CSS

Cascading StyleSheets (CSS) u Hojas de Estilo en Cascada es un lenguaje de diseño utilizado para definir y crear el estilo de visualización de un documento estructurado escrito en un Markup Language (ML). Creado en 1998, ha pasado por diferentes versiones corrigiendo errores (CSS1, CSS2, CSS2.1, CSS3.1) [19]. Utilizado principalmente para describir la presentación de páginas web, incluyendo colores, disposición de elementos y fuentes de los textos [20].

La separación de HTML y CSS busca mejorar la accesibilidad de la página, una mayor flexibilidad y control del estilo, así como permitir que varios documentos HTML compartan un mismo estilo usando una hoja de estilos. Esta separación también permite la descripción de la presentación de los elementos páginas web para diferentes tipos de dispositivos y tamaños de pantalla, y diferentes estilos de renderizado como en impresión o voz (haciendo uso de dispositivos o software que lo permitan) [20].

CSS utiliza una sintaxis simple y un conjunto de palabras clave en inglés para especificar el nombre de las propiedades de estilo. Una hoja de estilo CSS consiste en una serie de reglas, las cuales poseen selectores y declaraciones. Un selector indica a qué elemento o elementos se le aplicarán las declaraciones, las cuales definen la información del estilo [19].

2.3.5 JavaScript

JavaScript (JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript, que apareció en 1995. Es un lenguaje orientado a objetos, aunque puede funcionar como un lenguaje procedimental, dinámico y basado en prototipos. JavaScript es más conocido como el lenguaje de scripting de páginas web que permite el control del comportamiento de páginas web (animación de elementos de los elementos de la página, validación de valores de entrada en formularios...), aunque también es utilizado en otros entornos. JavaScript funciona en el lado del cliente, permitiendo mejoras en la interfaz de usuario y páginas web dinámicas [21].

2.3.6 AJAX

Asynchronous JavaScript And XML (AJAX) es una serie de técnicas de desarrollo web para crear aplicaciones web interactivas, que aparecieron en 1999. Las aplicaciones en desarrolladas en AJAX se ejecutan en el lado del cliente, es decir, en el explorador del usuario mientras se mantiene una comunicación asíncrona con el servidor en segundo plano. Ajax es la combinación de tecnologías existentes, en las que se encuentran XHTML (o HTML), CSS, The Document Object Model (DOM), XML, JavaScript, y la más importante, XMLHttpRequest object. Gracias a la combinación de estas tecnologías en el modelo Ajax, las aplicaciones son capaces de realizar recargas de elementos de la página web sin necesidad de recargar toda la página. Esto aumenta la velocidad de las páginas, así como la responsividad a las acciones de los usuarios [22].

2.3.7 SQLite

SQLite es un sistema de gestión de bases de datos relacional de dominio público originalmente lanzado en el año 2000. El sistema de gestión SQLite está contenido en una biblioteca escrita en C [23].

La biblioteca SQLite es una base de datos SQL enlazada directamente con el programa, siendo esta parte fundamental del mismo. SQLite lee y escribe directamente sobre ficheros ordinarios del disco. Una base de datos SQL completa está contenida en único fichero. Este diseño se consigue bloqueando todo el fichero de base de datos al principio de cada transacción de escritura, aunque la lectura si se puede realizar de forma simultánea. El fichero de base de datos es multiplataforma, se puede copiar entre arquitecturas de 32-bit y 64-bit o entre big-endian y little-endian [23].

2.3.8 Bootstrap

Bootstrap es un set de herramientas de código abierto y multiplataforma, creado en el 2011. Bootstrap es utilizado en el desarrollo con HTML, CSS y JS. Contiene plantillas elementos pres construidos, variables, iconos… Gracias a esto se facilita el desarrollo de páginas web [24].

3 Análisis

A lo largo de este capítulo se procederá a realizar un análisis del proyecto detallando el objetivo del proyecto, el alcance del proyecto, requisitos funcionales, requisitos no funcionales y casos de uso.

3.1 Objetivo

La aplicación web debe permitir la interacción con el motor de diálogo por parte de los usuarios, además debe permitir también la consulta de las diferentes conversaciones entre los usuarios y el chatbot por parte de los psicólogos, así como la gestión de dichas conversaciones.

3.2 Alcance

Inicialmente se definieron tres perfiles para la aplicación web; el perfil *paciente*, que permite a cualquier persona la interacción con el motor de diálogo, el *psicólogo*; que permite el acceso a la parte de administración y la consulta de conversaciones; y el *administrador*¸ el cual podrá realizar tarea más avanzadas como son la eliminación de conversaciones, la asignación de estas a *psicólogos, …*

Se enumerarán tanto los requisitos funcionales como los no funcionales, basados en las reuniones y necesidades definidas entre las partes involucradas.

3.3 Definición de requisitos

3.3.1 Requisitos funcionales

Responden principalmente a los requisitos de los usuarios y sus restricciones en tanto a los servicios que prestará la aplicación web. De momento existirán tres tipos de usuarios que podrán usar la web: psicólogos, administradores y pacientes.

- **RF01: Interactuar con el chatbot:**

  Los visitantes podrán mantener una conversación con el chatbot.

- **RF02: Iniciar de sesión:**

  Existirá una página de inicio de sesión para que los psicólogos y administradores puedan acceder a la sección de administración.

- **RF03: Consultar conversaciones:**

  Tanto los psicólogos como los administradores podrán acceder al historial de conversaciones mantenidas por los pacientes con el motor de diálogo.

- **RF04: Consultar de mensajes:**

  Tanto los psicólogos como los administradores podrán acceder a los mensajes de las distintas conversaciones mantenidas por los pacientes con el motor de diálogos.

- **RF05: Descargar conversaciones:**

  Tanto el administrador como los psicólogos podrán descargar un fichero CVS con el contenido de la conversación completa.

- **RF06: Eliminar conversaciones:**

  El administrador podrá eliminar conversaciones.

- **RF07: Asignar conversaciones:**

  El administrador podrá asignar conversaciones a psicólogos.

- **RF08: Crear de usuarios:**

  El administrador podrá crear nuevos usuarios, ya sean otros administradores o psicólogos.

  Estos usuarios poseen:

    - Nombre de usuario.
    - Contraseña.
    - E-mail.
    - Nombre.
    - Apellidos.
    - Permisos de usuario.

- **RF09: Editar usuarios:**

  El administrador podrá editar los diferentes campos de los usuarios:

    - Editar nombre de usuario.

- Editar contraseña.
- Editar e-mail.
- Editar nombre.
- Editar apellidos.
- Editar permisos.

3.3.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que se utilizan para analizar las diferentes propiedades del sistema, y no su comportamiento específico.

1. Rendimiento:

   La aplicación debe tener una respuesta rápida, tanto en la parte de interacción con el chat, para que los *pacientes*, no se sientan tentados a abandonar la conversación, como en la parte de la administración cuando se consulten las diferentes conversaciones.

2. Usabilidad

   Se busca que la aplicación web sea intuitiva: sencilla, clara, limpia, fácil de entender y de usar.

3. Adaptabilidad

   Se busca que la aplicación web sea adaptable, es decir, que la aplicación web se visualice correctamente independientemente del dispositivo utilizado, del sistema utilizado o del tamaño del dispositivo.

4. Soportabilidad

   La aplicación debe ser fácil de instalar y mantener.

5. Fiabilidad

   En caso de que la aplicación falle está debe asegurar la capacidad de recuperación.

6. Seguridad

   Únicamente los administradores o los psicólogos deberán tener acceso a las funcionalidades de administraciones de conversaciones, …

7. Escalabilidad

La aplicación debe permitir la posibilidad de añadir más usuarios, almacenamiento, nuevos módulos… sin empeorar la calidad del servicio de esta.

3.4 Casos de uso

En este parte del documento se mostrará el diagrama de casos de uso del proyecto desarrollado, *figura 3-1*, donde se mostrará de manera clara las diferencias entre los diferentes tipos de usuario de la aplicación web. Este diagrama es una representación de las distintas acciones que se deben realizar para que un usuario realice un proceso. En este diagrama se representan los diferentes actores involucrados en las diferentes acciones.

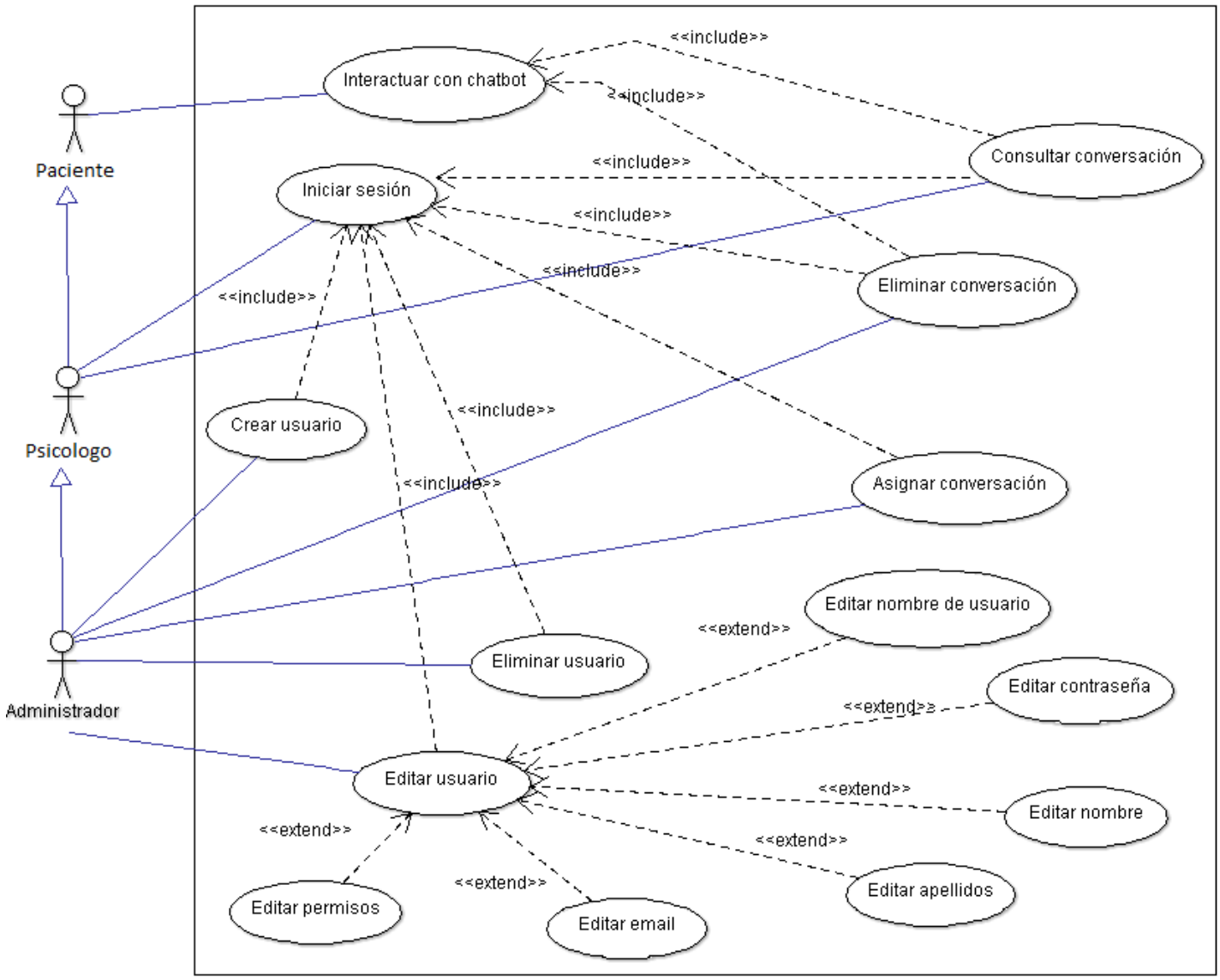

Figura 3-1: Diagrama de casos de uso

A continuación, detallaremos algunos de los casos de uso más representativos de la aplicación.

- **Caso de uso asignar conversación.**
  - **Actor primario:**

    Administrador.

  - **Interesados y objetivos:**

    1. Administrador: desea asignar una conversación a un psicólogo concreto.

  - **Precondiciones:**

    Debe haber al menos una conversación almacenada, debe existir el usuario del psicólogo al que se desea asignar la conversación. El administrador debe haber iniciado sesión correctamente.

  - **Garantía de éxito (Postcondiciones):**

    El administrador podrá ver dicha conversación asociada a un psicólogo. El psicólogo podrá ver la conversación asignada él.

  - **Escenario principal de éxito**:

    1. El administrador selecciona chats.
    2. El administrador selecciona la conversación deseada.
    3. El administrador selecciona el psicólogo.
    4. El administrador pulsa grabar.
    5. El sistema asigna la conversación al psicólogo.

  - **Extensiones (flujos alternativos):**

    No.

  - **Requisitos especiales:**

    1. Respuesta rápida del sistema a la hora de cargar las conversaciones y cuando se asignen estás.

  - **Frecuencia de ocurrencia:**
    Alta, se espera muchas conversaciones sean asignadas a distintos psicólogos.

- **Caso de uso crear usuario.**
  - **Actor primario:**

    Administrador.

  - **Interesados y objetivos:**

    1. Administrador: desea crear un usuario.

  - **Precondiciones:**

    El administrador debe haber iniciado sesión correctamente.

  - **Garantía de éxito (Postcondiciones):**

    El administrador podrá ver dicho usuario creado. El usuario deberá poder iniciar sesión correctamente.

  - **Escenario principal de éxito**:

    1. El administrador selecciona usuarios.
    2. El administrador pulsa añadir usuario.
    3. El administrador introduce el nombre de usuario y contraseña deseados.
    4. El administrador pulsa grabar.
    5. El administrador selecciona los permisos deseados para dicho usuario.
    6. El administrador pulsa guardar.

  - **Extensiones (flujos alternativos):**

    No.

  - **Requisitos especiales:**

    Ninguno.

  - **Frecuencia de ocurrencia:**

    Baja, se espera que se creen pocos usuarios en la aplicación.

4 Diseño

4.1 Introducción

En este capítulo veremos una introducción a la arquitectura que tendrá la aplicación web, hablando de manera general de los elementos que componen dicha arquitectura y de las tecnologías necesarias para el desarrollo de la aplicación.

4.2 Esquema de arquitectura

Después de definir los requisitos tanto funcionales como no funcionales de la aplicación web vamos a ver de manera simple la arquitectura que dará soporte a la aplicación. La arquitectura de la aplicación está dividida en dos partes:

- **Front-end**

  Esta capa se encarga de la interacción de los diferentes usuarios con la aplicación y de la representación y estilización de los datos de la aplicación. En esta capa estarán incluidas todas las tecnologías que serán utilizadas en los navegadores web. Algunas de las tecnologías más utilizadas para satisfacer estas necesidades son HTML5, CSS, Bootstrap, JavaScript, Ajax, …

- **Back-end**

  Capa en la que se realizarán los procesos para generar las respuestas a las entradas de los usuarios, así como el almacenamiento, acceso, y gestión de los datos necesarios. En las tecnologías implicadas en esta capa para poder realizar dichos procesos están incluidas un framework web, un gestor de bases de datos para y la API Watson Assistant v1.

En la *figura 4-1* podemos ver a grandes rasgos los diferentes elementos que componen la arquitectura de la aplicación. En el front-end estará el cliente móvil o web para la visualización, en esto también entran algunas partes del framework web que se encarga de encargar de representar la información para que pueda ser posteriormente visualizada. En el back-end tenemos el framework web que se encarga de procesar las distintas peticiones de los usuarios, de la comunicación con la base de datos y de la comunicación con Watson Assistant a través de su API.

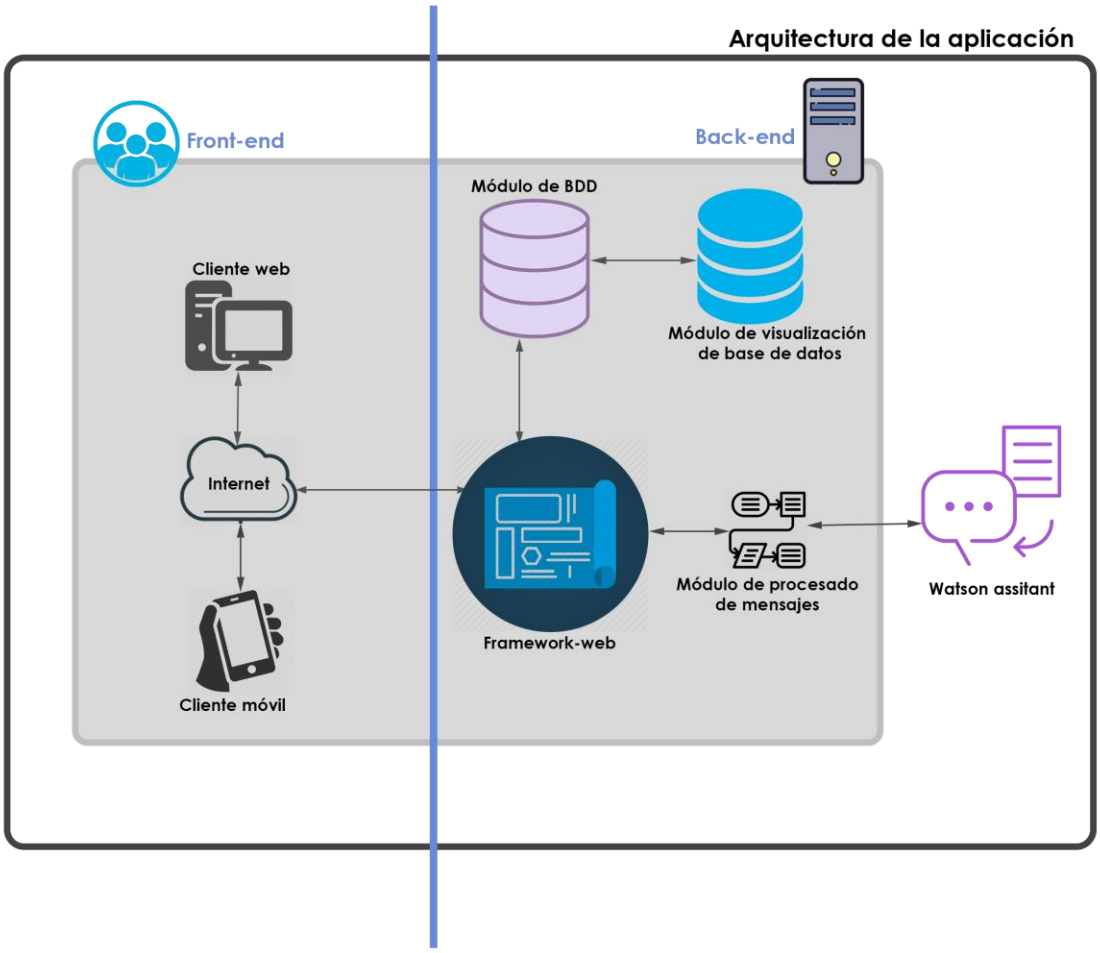

Figura 4-1: Arquitectura de la aplicación

4.3 Elementos de la arquitectura

Después de haber visto la arquitectura de la aplicación en la *figura 4-1*, vamos a definir cada uno de los elementos que componen las partes anteriormente mencionadas:

- **Framework web**

  Este módulo es primordial en la arquitectura ya que es el componente que hace de enlace entre todos los demás elementos que componen la aplicación. Se encarga de la representación de las páginas web, al igual que de todos los elementos que estén asociadas a estas. Algunos de los frameworks más importantes son Angular, Nodejs, Flask, Django, …

- **Módulo de base de datos**

  En este módulo reside el mayor peso de la aplicación ya que es donde se encuentran almacenados los diferentes usuarios de la aplicación y las conversaciones que mantengan los usuarios con el motor de diálogo. El número de conversaciones que deberá almacenar la aplicación será elevado por lo que se necesitará encontrar una tecnología que nos permita un fácil y rápido acceso a la información y, además, dicha tecnología nos debe permitir una fácil integración con el framework utilizado. Existen gran cantidad de opciones que en función de las necesidades que se deseen como son: PostgreSQL, MySQL, SQLite, …

- **Módulo de visualización de base de datos**

  En este caso existen muchas herramientas que permiten la visualización de la información de la base de datos de una manera rápida e intuitiva. Gracias a este tipo de herramientas se facilita el desarrollo de las aplicaciones web usando estas para comprobar la funcionalidad de nuestra aplicación y también que los datos se almacenen de manera correcta. Una de estas herramientas podría ser Django Admin.

- **Módulo de procesado de mensajes**

  Este módulo es el encargado de recibir y enviar los mensajes intercambiados entre el usuario y el motor de diálogo, haciendo de puente entre el framework web y Watson Assistant.

5 Desarrollo

5.1 Introducción

En este capítulo veremos de manera más detallada los elementos que componen la arquitectura de nuestra aplicación, es decir, veremos en profundidad las diferentes tecnologías utilizadas. También veremos una explicación detallada sobre las diferentes funcionalidades, el flujo de navegación y el diseño de la aplicación.

5.2 Arquitectura de Chatbot LISA

Una vez realizado el análisis de la arquitectura y las diferentes tecnologías que pueden estar implicadas en las partes de esta, procedemos a definir las teologías utilizadas en la implementación de dicha arquitectura:

- **Front-end**

  En esta parte de la arquitectura que se encarga de gestionar las diferentes interacciones de los usuarios con la aplicación utilizaremos el framework web Django; los lenguajes de programación Python, y JavaScript; y HTML5, CSS, AJAX y Bootstrap pará el diseño web.

- **Back-end**

  En esta capa, atendiendo a las necesidades requeridas, utilizaremos Django para el procesado de las peticiones de páginas web de los usuarios, SQLite para el almacenamiento de la información, Django Admin para la visualización de esta, el lenguaje de programación Python y la API Watson Assistant v1, para el intercambio de información con el chatbot.

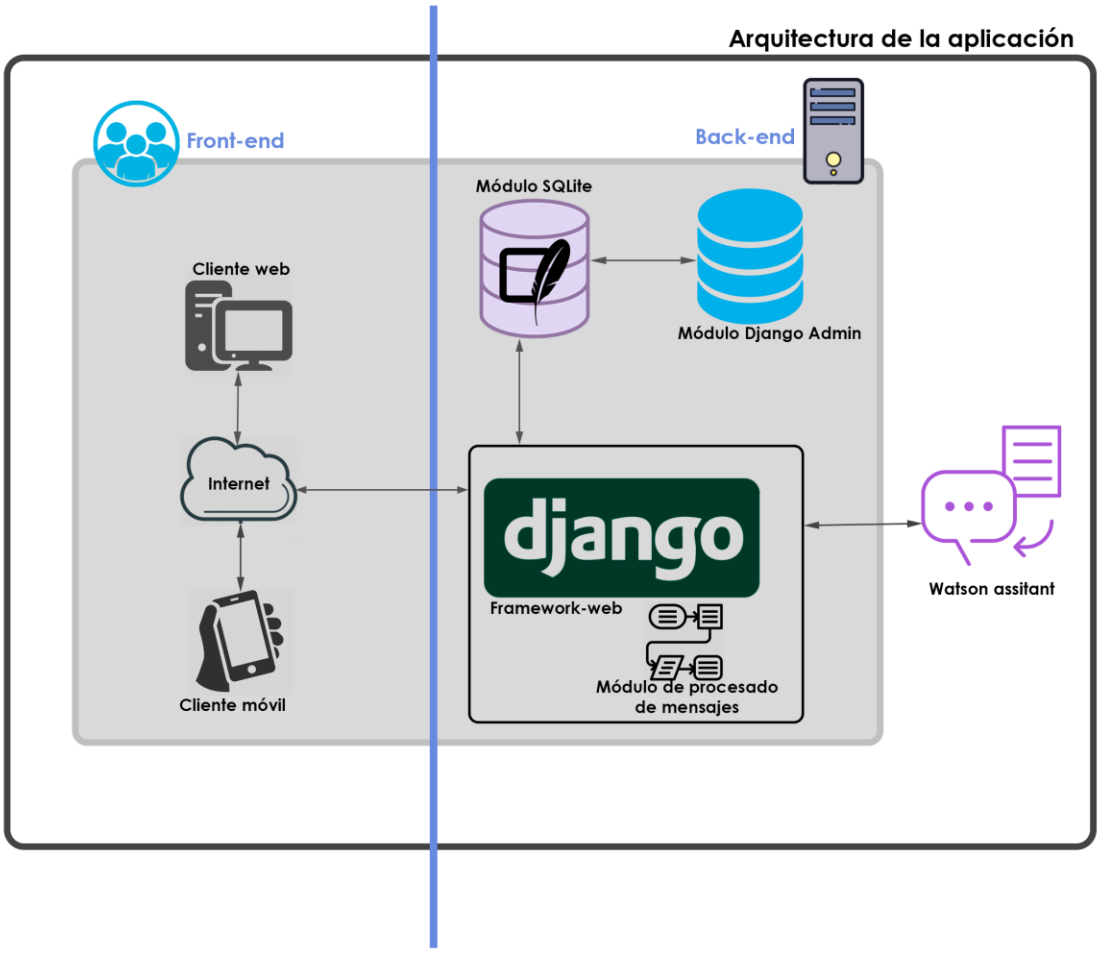

Figura 5-1: Arquitectura de la aplicación con las tecnologías específicas

5.3 Elementos del Chatbot LISA

Ahora vamos a detallas cada uno de los elementos que componen la arquitectura mostrada en la *figura 5-1*:

- **Django y módulo de procesado de mensajes:**

  Haciendo uso del Modelo-Vista-Plantilla (MTV), Django se encarga de la presentación de la página web, sus diferentes elementos, así como del acceso a la información contenida en la base de datos. También se encarga de la lógica de funcionamiento de la aplicación y del procesado de la información enviada por los usuarios, es decir, en Django está integrada la funcionalidad para la comunicación con Watson Assistant.

- **Módulo SQLite**

  Como mencionamos anteriormente este módulo es crítico para el funcionamiento de nuestra aplicación ya que aquí es donde se almacenarán

todas las conversaciones y los diferentes usuarios de la aplicación. Se utilizará la versión SQLite3. La integración de SQLite con Django es realmente sencilla, ya que este nos provee las herramientas para la construcción este tipo de base de datos.

En la *figura 5-2* vemos la estructura que tendrá dicha base de datos.

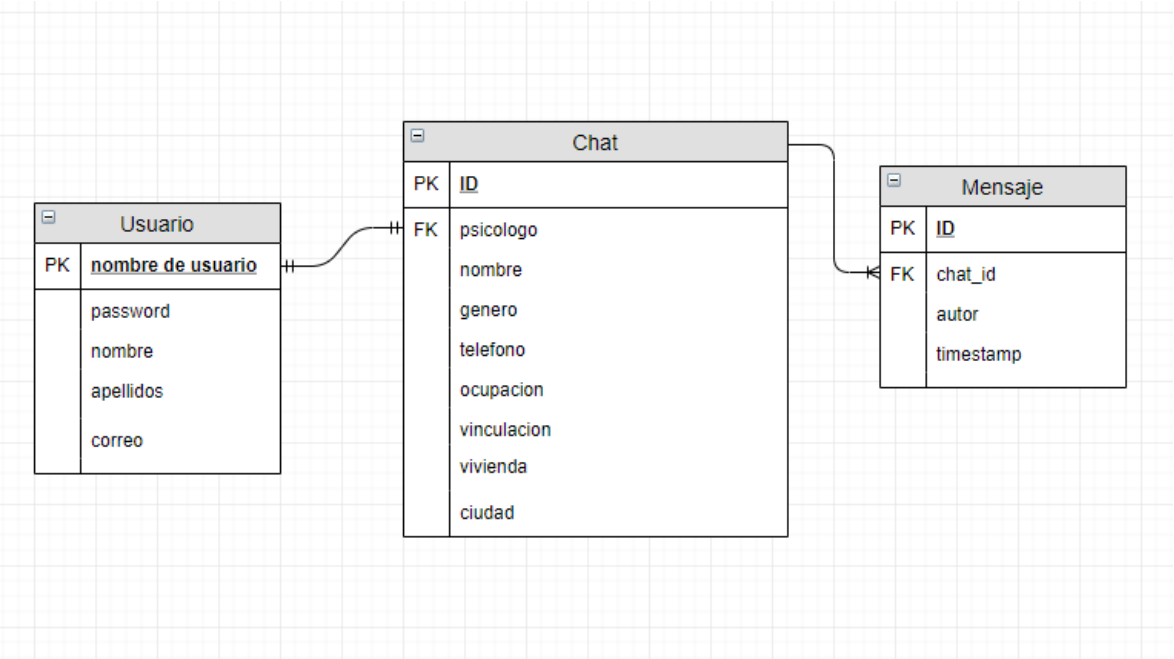

Figura 5-2: Diagrama entidad relación de la base de datos

- **Módulo Django Admin**

  Utilizado para la administración de toda la información de la base de datos.

5.4 Intercambio de información entre usuarios y Watson Assistant

En este apartado detallaremos la implementación del módulo de procesado de mensajes y la funcionalidad asociada a este.

En la *figura 5-3* vemos la conexión a Watson Assistant para ello utilizando su API.

```
import json
from watson_developer_cloud import AssistantV1


WORKSPACE_ID = '[redacted]'
assistant = AssistantV1(
    version='2018-09-20',
    username='[redacted]
    password='[redacted]',
    url='https://gateway.watsonplatform.net/assistant/api')
```

Figura 5-3: Conexión a Watson Assistant

En la *figura 5-4* podemos ver el procedimiento que sigue nuestro módulo de procesamientos de mensaje para enviar y recibir mensajes de Watson Assistant, utilizando una variable de sesión de Django para almacenar la conversación mantenida en caso de que haya que almacenar esta. Además, aquí es donde se encuentra la implementación de escritura en base de datos de las conversaciones. Los momentos en los que se deberán guardas las conversaciones son:

- Cuando el usuario envíe el mensaje:
    - *"Lisa, quiero terminar"*.
- Cuando Watson Assistant llegué a un nodo final, es decir, cuando responda:
    - *"Ya hemos terminado"*.
    - *"Gracias por ponerte en contacto con nosotros"*.

En cualquiera de los casos mencionados anteriormente el campo de envío de mensajes de la página web deberá quedar bloqueado.

```python
mensajes_finales = ["Ya hemos terminado","Gracias por ponerte en contacto con nosotros"]
@csrf_exempt
def apiWatson(request=None):
    context = {}
    flagOptions = False
    final = False

    mensaje =  json.loads(request.body)
    try:
        if mensaje['context']:
            context = mensaje['context']
    except:
        pass

    response = assistant.message(
        workspace_id= WORKSPACE_ID,
        input = {'text': mensaje['input']['text']},
        context = context,
    ).get_result()

    #respuestas de LISA
    respuestaWatson = response["output"]["text"]
    for r in response["output"]["generic"]:
        if 'options' in r:
            for o in r['options']:
                respuestaWatson.append(o['label'])


    interaccion = {"mensaje":  mensaje['input']['text'], "respuesta": respuestaWatson ,
    "fecha": datetime.now().strftime('%Y-%m-%dT%H:%M:%S.%f')}

    if 'chat' not in request.session or not request.session['chat']:
        request.session['chat'] = [interaccion]
    else:
        request.session['chat'].append(interaccion)
    request.session.modified = True

    for r in respuestaWatson:
        for m in mensajes_finales:
            if (m.lower() in r.lower()):
                final = True
                break

    #Si el usuario decide terminar
    if (("LISA, terminar").lower() in mensaje['input']['text'].lower()) or (final):

        chat = Chat.objects.create(nombre=response["context"]["nombre"], edad=response["context"]["edad"],
            genero=response["context"]["genero"], telefono=response["context"]["telefono"],
            ocupacion=response["context"]["ocupacion"],
            vinculacion=response["context"]["vinculacion"],
             vivienda=response["context"]["vivienda"], ciudad=response["context"]["ciudad"])

        for interacc in request.session['chat']:
            cuerpo = ""
            if mensaje != '':
                Mensaje.objects.create(chat_id = chat, autor = response["context"]["nombre"],
                    cuerpo= interacc["mensaje"], timestamp=datetime.now().strptime(interacc["fecha"],'%Y-%m-%dT%H:%M:%S.%f'))

            for respuesta in interacc["respuesta"]:
                cuerpo+=respuesta
            Mensaje.objects.create(chat_id = chat, autor = "watson", cuerpo= cuerpo,
                timestamp=datetime.now().strptime(interacc["fecha"],'%Y-%m-%dT%H:%M:%S.%f'))

    return  HttpResponse(json.dumps(response), content_type="application/json")
```

Figura 5-4: Código para el procesado de mensajes

5.5 Interfaz web

5.5.1 Chatbot LISA

En esta página se podrán enviar y recibir mensajes para mantener una conversación con el chatbot de Watson Assistant, *figura* 5-5.

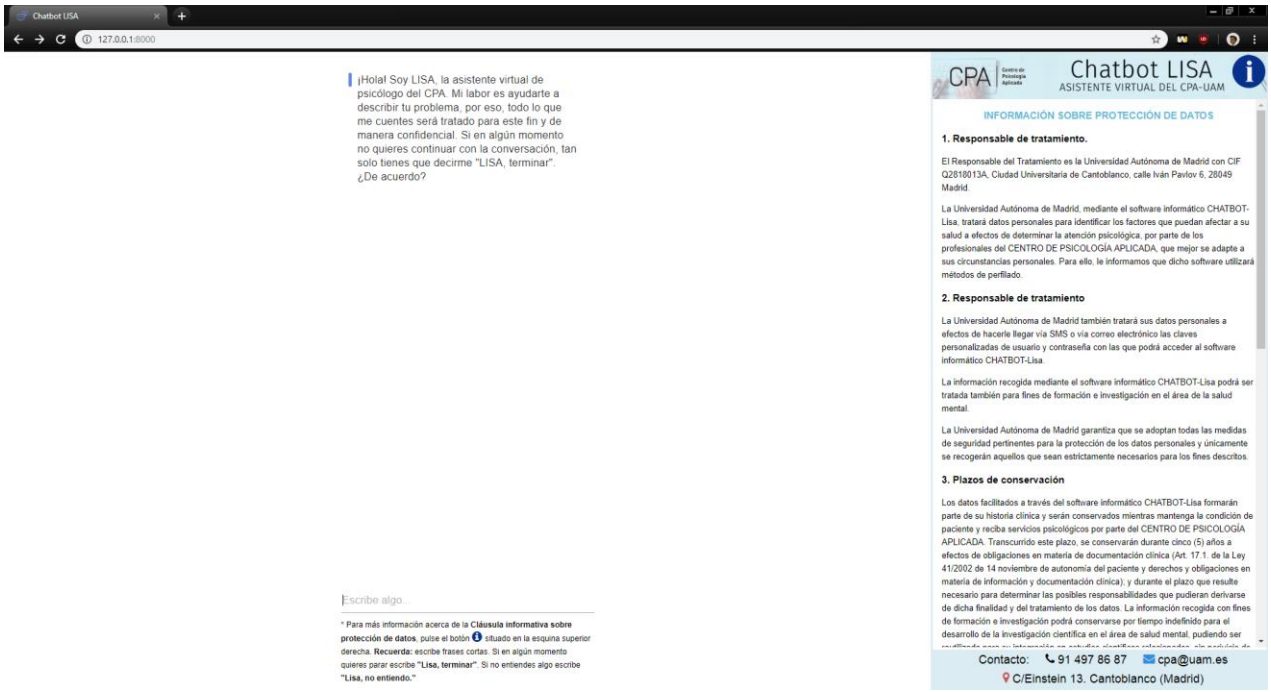

Figura 5-5: Pantalla del chatbot

5.5.2 Django Admin

Interfaz que se podrá controlar y visualizar todos los datos almacenados en la base de datos.

5.5.2.1 Django Admin Login

En la *figura 5-6* se muestra la pantalla de inicio de sesión en la página de administración de Django. Una vez se haya realizado el inicio de sesión correctamente, el usuario será redirigido a la página principal de administración.

Figura 5-6: LogIn Django Admin

Dependiendo del usuario que inicie sesión en la página de administración de Django esta tendrá un contenido u otro.

5.5.2.2 Django Admin página principal

En la *figura 5-7* podemos ver el contenido de la página principal del administrador de Django cuando se ha iniciado sesión siendo administrador.

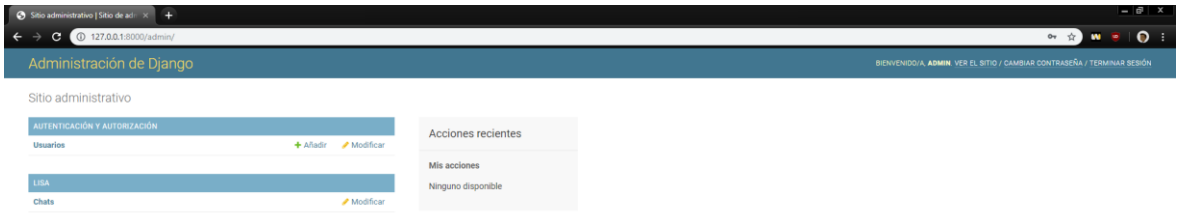

Figura 5-7: Página principal Django Admin si se es administrador

En la *figura 5-8* podemos ver el contenido de la página principal del administrador de Django cuando se ha iniciado sesión siendo psicólogo.

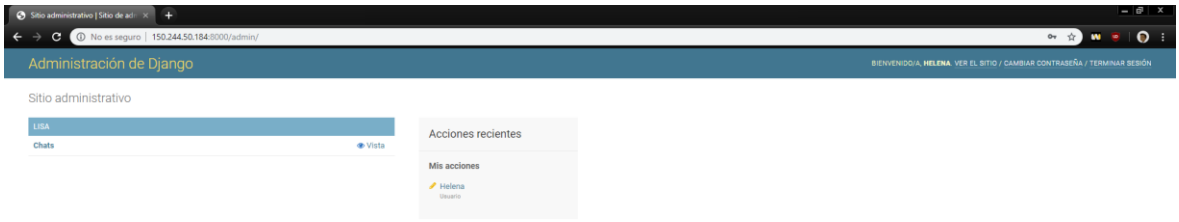

Figura 5-8: Página principal Django Admin si se es psicólogo

5.5.2.3 Django Admin página de chats

En la *figura 5-9* podemos ver el contenido de la página de chats tanto para los psicólogos como para el administrador, la diferencia en este caso está en que el administrador posee una opción para eliminar las conversaciones seleccionadas en el botón desplegable.

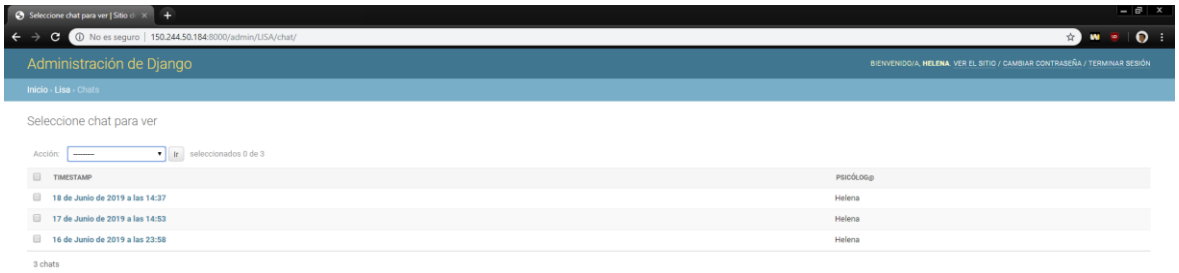

Figura 5-9: Página de chats Django

5.5.2.4 Django Admin página de conversación

En la *figura 5-10* podemos ver la visualización de una de las conversaciones cuando se ha iniciado sesión como administrador. Dentro del desplegable el administrador podrá escoger a que psicólogo desea asignar dicha conversación.

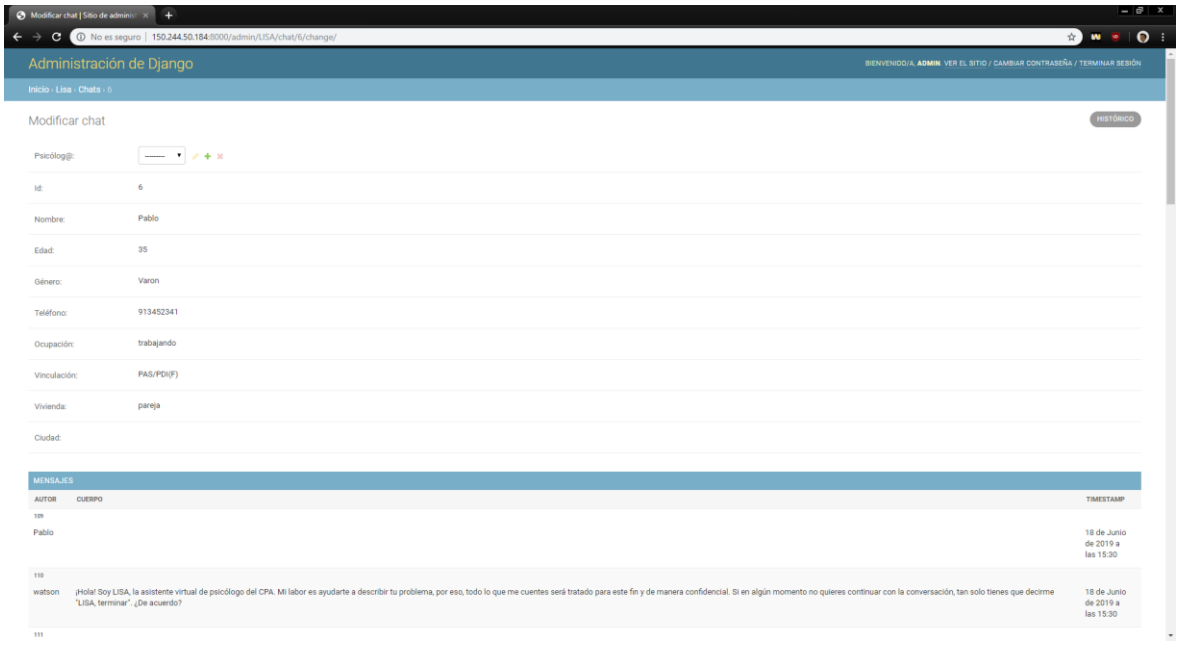

Figura 5-10: Django Admin página de conversación siendo administrador

En la *figura 5-11* podemos ver la visualización de una de las conversaciones cuando se ha iniciado sesión como psicólogo.

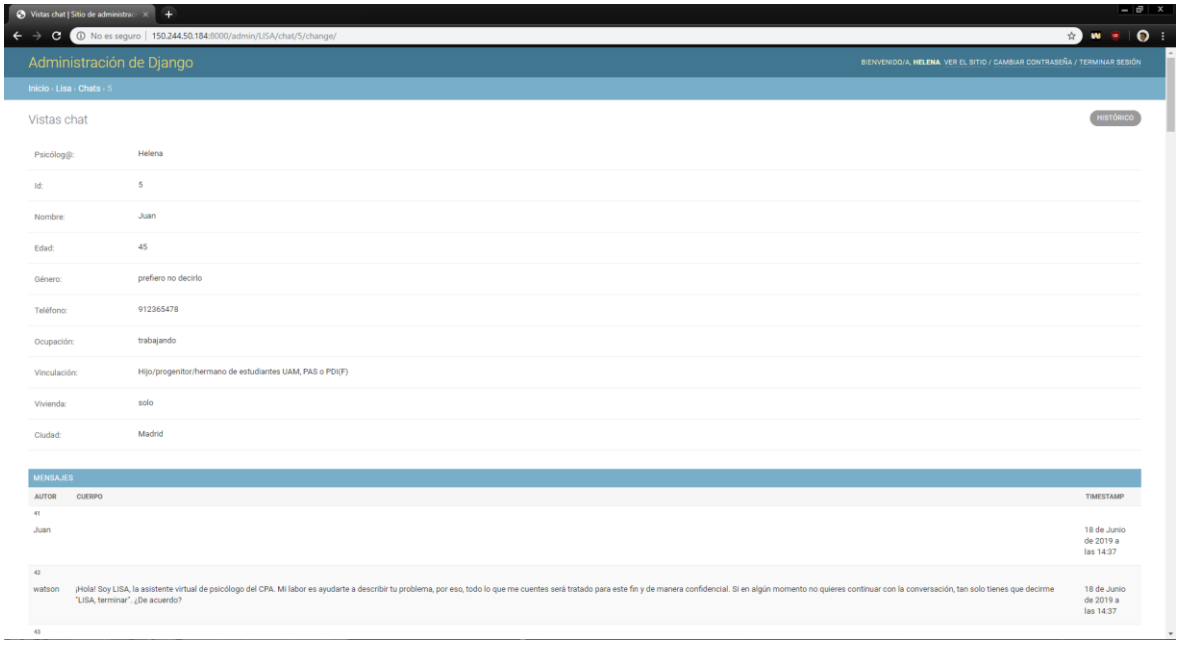

Figura 5-11: Django Admin página de conversación siendo psicólogo

En la *figura 5-12* podemos ver la visualización de una de la página de usuarios de Django Admin, esta parte solo es accesible para el administrador.

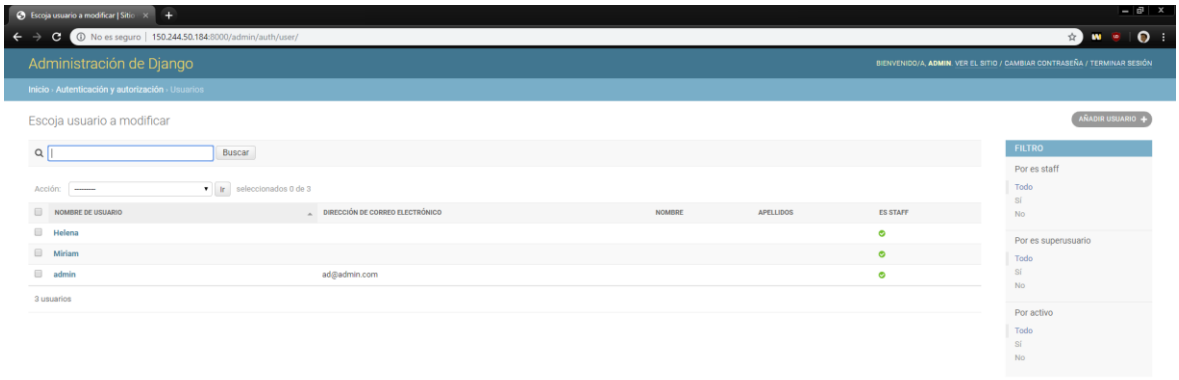

Figura 5-12: Django Admin Página de usuarios

En la *figura 5-13* podemos ver la visualización de una de la página de creación usuarios de Django Admin, esta parte solo es accesible para el administrador.

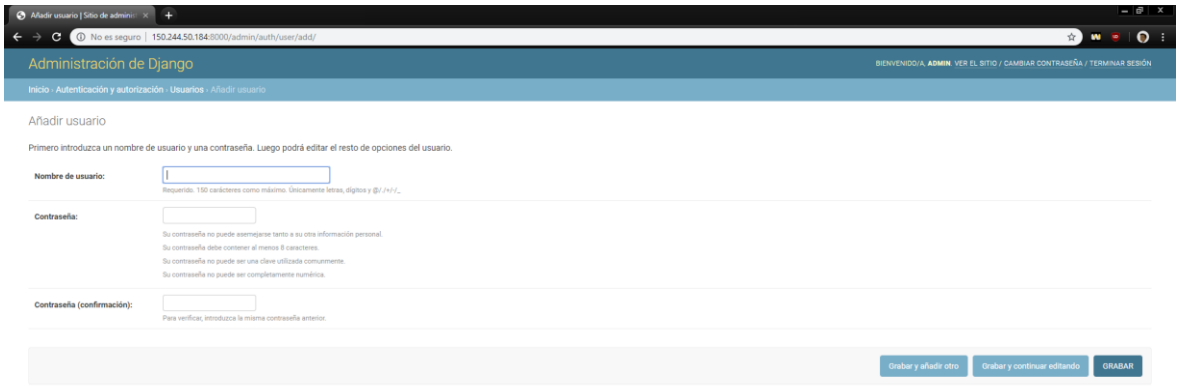

Figura 5-13: Django Admin creación de usuarios

5.6 Interfaz web responsiva

En las *figuras 5-14, figura 5-15 y figura 5-16* podemos observar el comportamiento de la web que contiene la interacción con el chatbot en distintos dispositivos, cumpliendo con esto la idea de diseño responsivo.

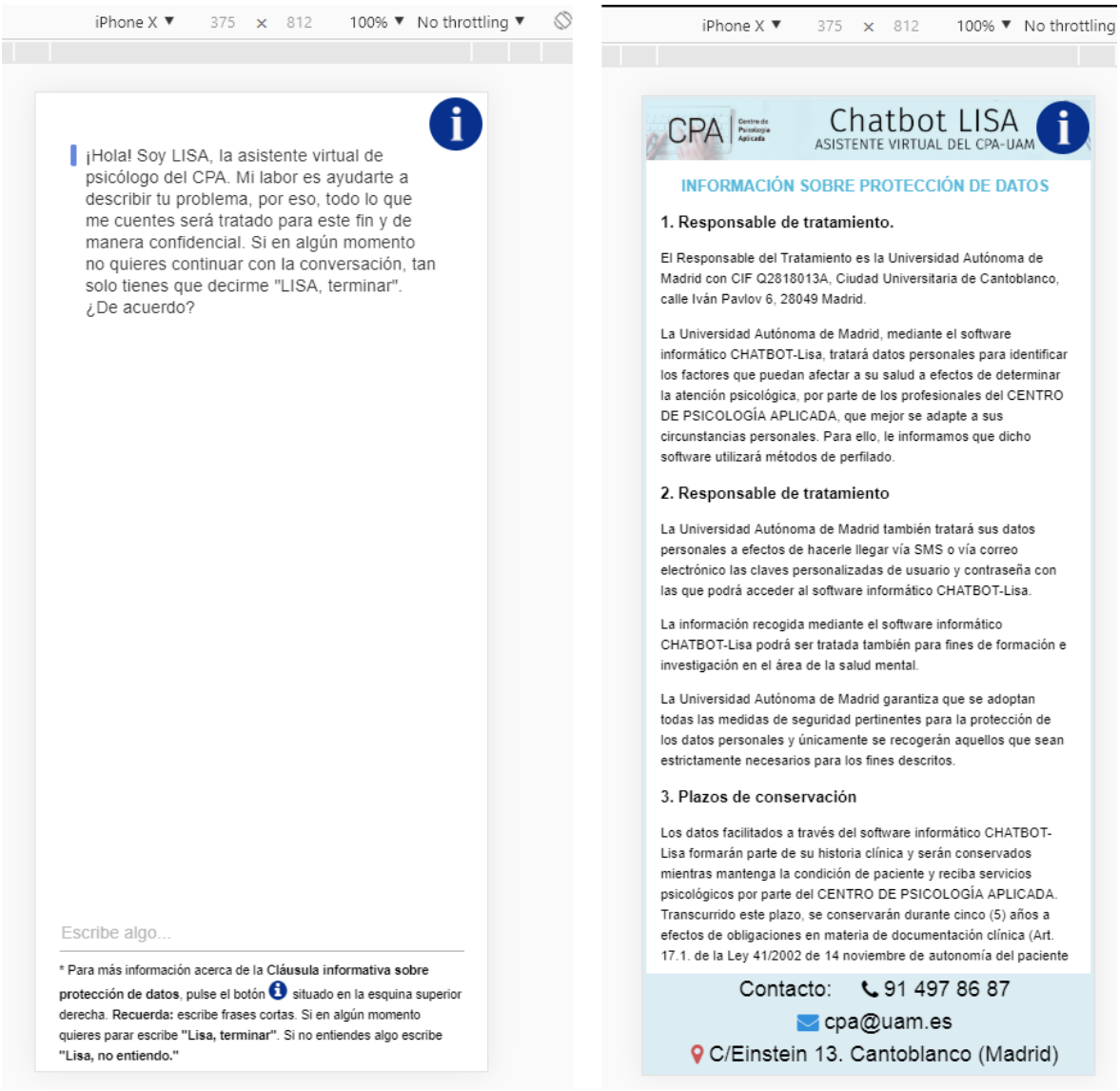

Figura 5-14: Pantallas chatbot en móvil

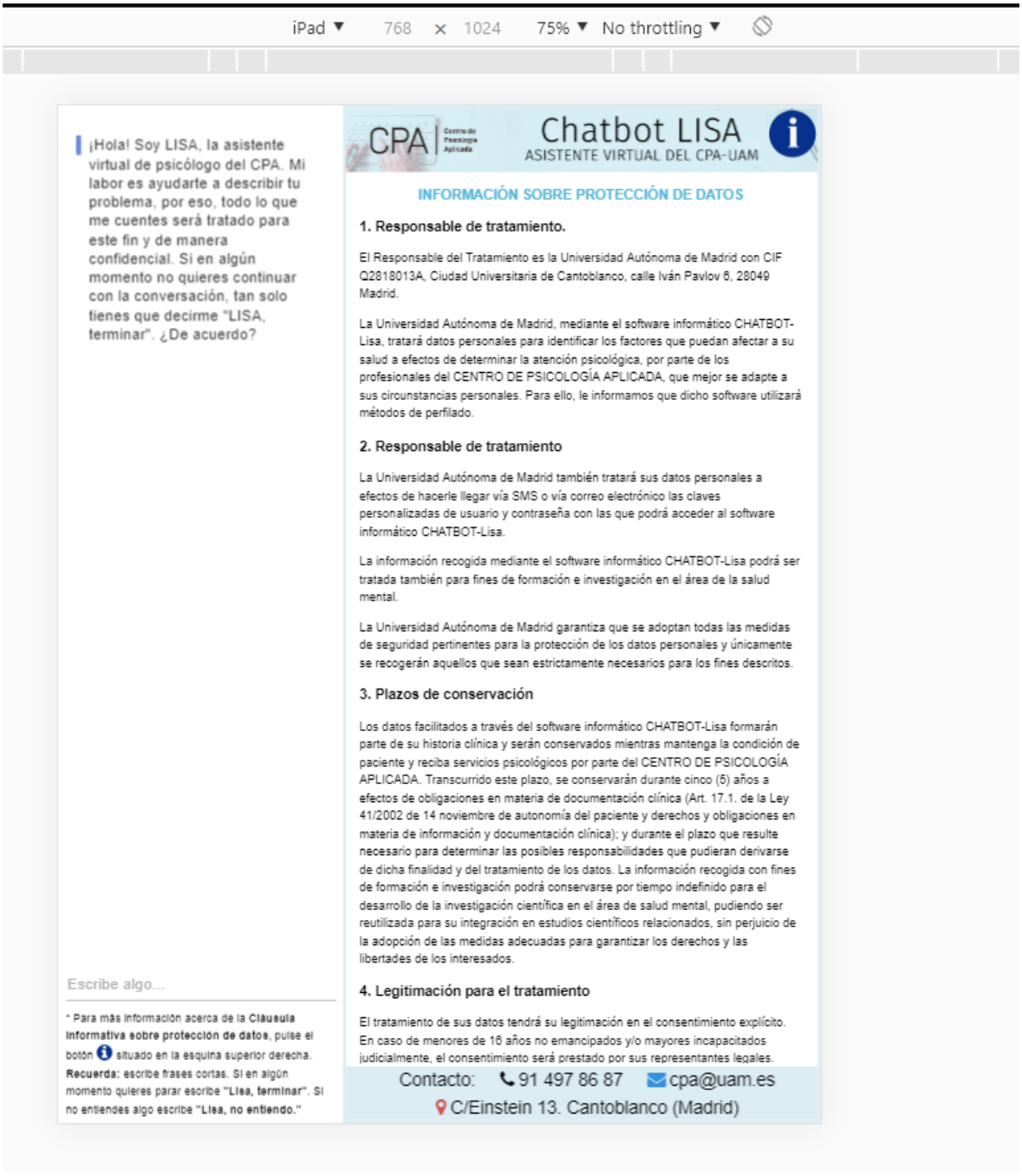

Figura 5-15: Pantalla chabot en tablet

Figura 5-16: Pantalla chatbot en resolución diferente

6 Integración y pruebas

6.1 Integración

Para la integración de los diversos módulos que componen la aplicación usamos Django como estructura principal del proyecto, añadiendo a este el código para integrar la API de Watson Assistant y las diferentes bibliotecas de programación necesarias para manejar la comunicación don dicha API. También se añadieron a Django las diferentes herramientas de base de datos SQLite, así como el administrador de estas.

6.2 Pruebas

La gran parte de las pruebas se han centrado en la comprobación en primer lugar de las diferentes tecnologías.

Primero se comprobó que la configuración y elementos de Django estuvieran correctamente. Para esto se creó una página web sencilla, y se comprobó que las peticiones a Django mostrarán dicha página correctamente.

Después se probó la integración de Django y la API de Watson Assistant, probando que la conexión y el envío de mensajes a Watson Assitant, así como la respuesta de este, fuesen correctos.

Una vez comprobada la integración Django y la API de Watson, procedimos a probar la integración de la página web en la que se muestra el chat, realizando peticiones desde está a Django y comprobando que la visualización de las respuestas fuese correcta.

Después se probó la integración completa de toda la aplicación. Una vez comprobado la correcta integración de todos los componentes mencionados anteriormente se procedió a probar la integración de estos con la base de datos SQLite y el módulo de administración de base de datos Django. Para comprobar esto se procedió a mantener conversación con el chatbot, ver que estas se almacenaban correctamente, accediendo al módulo de administración de bases de datos. También para probar esto se crearon nuevos *psicólogos* (usuarios), comprobando el correcto acceso, descarga y visualización de las conversaciones en el módulo de administración. Y por último se probó que la asignación de conversaciones a los diferentes usuarios se realizará correctamente.

Además de las pruebas realizadas en el entorno de desarrollo, las mencionadas anteriormente, se fueron realizando pruebas en un servidor, proporcionado por el Laboratorio de lingüística informática, en las que se fueron simulando y probando algunas fases de la posible instalación. Con estas pruebas de fueron identificando errores y mejoras por parte de el CPA.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

Tras haber finalizado el TFG y habiendo analiza los resultados obtenido, podemos observar como los objetivos deseados han sido alcanzado en su gran mayoría.

Al haber realizado un TFG multidisciplinar y habiendo tantas personas involucradas he podido observar, tal y como hemos ido viendo a lo largo de la carrera, que la fase de análisis es un momento fundamental dentro del desarrollo de software. Si esta fase no se lleva a cabo correctamente o se hace de manera pobre puede llevar a confusiones futuras. Ligando esto con la realización del TFG he podido darme cuenta de que no se consolido bien la fase anteriormente mencionada ya que en gran parte de las reuniones se iban cambiando algunos de los requisitos funcionales de la aplicación, así como añadiendo otros. Este pequeño problema no fue impedimento para la finalización del TFG, pero si ha sido una gran lección aprendida.

Por último, me gustaría añadir que la realización de este TFG me ha aportado conocimientos de algunas disciplinas que desconocía, la experiencia de cómo sería elaborar un proyecto de software, las distintas fases por las que debe pasar este y problemas que pueden ir surgiendo en su desarrollo.

7.2 Trabajo futuro

A este TFG aún le quedan algunas cosas por mejorar, que por falta de tiempo no han podido ser implementadas:

- Añadir el botón de descargar las conversaciones en la misma fila de la conversación. Mejorando con esto la usabilidad de la aplicación.

- Mejorar la interfaz completa de Django Admin, proporcionándole un aspecto más adecuado y nombres correctos.

- En caso de que el volumen de usuarios de la aplicación crezca en la parte de administración/psicólogos se podría modificar la tecnología de base de datos utilizada por una que soporte un mayor número de usuarios simultáneos.

- Incluir un servidor de correo y adecuar la funcionalidad para enviar un correo a los psicólogos en caso de que algún potencial paciente esté en peligro y se lo haya comunicado al chatbot.

Referencias

[1] Fundeu BBVA. (2019). *«Chatbot», neologismo válido.* [online] Available: https://www.fundeu.es/recomendacion/chatbot-neologismo-valido/

[2] S.N. (2019). *JSON*. [online] Available: https://www.json.org/

[3] S.N. (2019). *¿Qué es una API y para qué sirve?* [online] Available : https://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html

[4] S.N. (2019). *What is REST? | Codecademy*. [online] Available: https://www.codecademy.com/articles/what-is-rest

[5] Christensson, P. (2011). *Markup Language Definition*. [online] Available: https://techterms.com/definition/markup_language

[6] S.N. (2019). *Introducción a XML* [online] Available: https://developer.mozilla.org/es/docs/Web/XML/Introducci%C3%B3n_a_XML

[7] Galindo-Haro, J. (2008). *Diseño e implementación de un marco de trabajo (framework) de presentación para aplicaciones JEE*. [online] Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/876/1/00765tfc.pdf

[8] Hansson, K. (2005). Trees Versus Bytes. [online] Available: http://central.kaserver5.org/Kasoft/Typeset/JavaTree/index.html

[9] Gerpe Lazzarini, P. (2017). *GUIA COMPLETA: crear un chatbot para un banco con Watson e integrarlo a Facebook Messenger sin…*. [online] Planeta Chatbot : todo sobre los Chatbots, Voicebots e Inteligencia Artificial. Available: https://planetachatbot.com/guia-completa-crear-un-chatbot-para-un-banco-con-watson-e-integrarlo-a-facebook-messenger-sin-c-346d7ca17131

[10] López Mancisidor, A. (2017). *¿Qué comemos hoy? Cocina con Watson en 10 min*. [online] Available: https://www.ibm.com/blogs/think/es-es/2017/06/29/cocinando-con-watson-conversation/?cm_mmc=OSocial_Blog-_-Developer_Productivity-_-ES_ES-_-Ana+L%F3pez&cm_mmca1=000022PX&cm_mmca2=10005424

[11] Python Software Foundation (2019). *History and License — Python 3.7.3 documentation*. [online] Available: https://docs.python.org/3/license.html

[12] Venners, B. (2019). *The Making of Python*. [online] Artima.com. Available: https://www.artima.com/intv/pythonP.html

[13] Caymans (2019). *PYTHON (informática) | Qué es, para qué sirve y caracteristicas*. 247 Tecno. [online] Available: http://247tecno.com/python-para-que-sirve-por-que-usarlo/

[14] Gibson, C. (2019). *The Web framework for perfectionists with deadlines | Django*. [online] Available: https://www.djangoproject.com/

[15] Holovaty, A. y Kaplan-Moss, J. (2019). *El libro de Django 1.0*. [online] Available: https://uniwebsidad.com/libros/django-1-0/capitulo-5/el-patron-de-diseno-mtv

[16] Infante-Montero, S. (2019). *Curso Django: Entendiendo como trabaja Django*.Maestros del Web. [online] Available: http://www.maestrosdelweb.com/curso-django-entendiendo-como-trabaja-django/

[17] Franganillo, J. (2011). *Html5: el nuevo estándar básico de la Web.* Anuario ThinkEPI. [online] Available: https://franganillo.es/html5.pdf

[18] Martínez, A. (2019). *HTML5*. Documentación web de MDN. [online] Available: https://developer.mozilla.org/es/docs/HTML/HTML5

[19] Bos, B. (2011). *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. [online] Available: https://www.w3.org/TR/2011/REC-CSS2-20110607/ [Accessed 17 Jun. 2019].

[20] S.N. (2019). *Learn to style HTML using CSS.* [online] Available: https://developer.mozilla.org/en-US/docs/Learn/CSS

[21] S.N. (2019). *About JavaScript.* [online] Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript [Accessed 17 Jun. 2019]

[22] S.N. (2019). *Ajax*. [online] Available: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX [Accessed 17 Jun. 2019].

[23] S.N. (2019). *About SQLite*. [online] Available: https://www.sqlite.org/about.html [Accessed 18 Jun. 2019].

[24] S.N. (2019). *Bootstrap*. [online] Available: https://getbootstrap.com/

Glosario

API	Application Programming Interface
TFG	Trabajo de fin de grado
RF	Requisito functional
JS	JavaScript
Ajax	Asynchronous JavaScript And XML
API Restful	API que utiliza peticiones HTTP como GET, PUT, POST y DELETE
SQL	Strructured Query Language
MTV	Model Template View
URL	Uniform Resource Locator

Anexos

A Ejemplo respuesta JSON de Watson Assistant


```
{
 "intents": [],
 "entities": [
  {
   "entity": "ocupacion",
   "location": [
    0,
    10
   ],
   "value": "estudiando",
   "confidence": 1
  }
 ],
 "input": {
  "text": "Estudiante"
 },
 "output": {
  "generic": [
   {
    "time": 1250,
    "typing": true,
    "response_type": "pause"
   },
   {
    "response_type": "text",
    "text": "\u00bfTienes alguna relaci\u00f3n con la Universidad Aut\u00f3noma de Madrid?"
   },
   {
    "title": "Selecciona una de las siguientes opciones:",
    "options": [
     {
      "label": "Estudiante UAM",
      "value": {
       "input": {
        "text": "Estudiante UAM"
       }
      }
     },
     {
      "label": "AlumniUAM* cursando estudios fuera de la UAM o en paro laboral",
      "value": {
       "input": {
        "text": "AlumniUAM* cursando estudios fuera de la UAM o en paro laboral"
```

```
          }
         }
        },
        {
         "label": "AlumniUAM* trabajando",
         "value": {
          "input": {
           "text": "AlumniUAM* trabajando"
          }
         }
        },
        {
         "label": "PAS/PDI(F)",
         "value": {
          "input": {
           "text": "PAS/PDI(F)"
          }
         }
        },
        {
         "label": "Hijo/padre/madre/hermano de estudiantes UAM, PAS o PDI(F)",
         "value": {
          "input": {
           "text": "Hijo/progenitor/hermano de estudiantes UAM, PAS o PDI(F)"
          }
         }
        },
        {
         "label": "Ninguna de las anteriores",
         "value": {
          "input": {
           "text": "Ninguna de las anteriores"
          }
         }
        }
       ],
       "description": "*Solo ser\u00e1 v\u00e1lido en sus modalidades AlumniUAM+, AlumniUAM+ Plus y Amigo de la UAM",
       "response_type": "option"
      }
     ],
     "text": [
      "\u00bfTienes alguna relaci\u00f3n con la Universidad Aut\u00f3noma de Madrid?",
      "Estudiante UAM",
      "AlumniUAM* cursando estudios fuera de la UAM o en paro laboral",
      "AlumniUAM* trabajando",
      "PAS/PDI(F)",
      "Hijo/padre/madre/hermano de estudiantes UAM, PAS o PDI(F)",
      "Ninguna de las anteriores"
     ],
```

```
  "nodes_visited": [
    "node_16_1538480554968",
    "node_3_1541521658106"
  ],
  "log_messages": []
},
"context": {
  "conversation_id": "1b58b71e-08eb-4c15-b064-64c269798382",
  "system": {
    "_node_output_map": {
      "Welcome": {
        "1": [
          0
        ]
      },
      "handler_7_1557419119801": [
        0,
        0
      ],
      "node_1_1537786506506": {
        "1": [
          0
        ]
      },
      "node_46_1537871607887": {
        "1": [
          0,
          0,
          1
        ]
      },
      "node_11_1537787086358": {
        "1": [
          0,
          1,
          0
        ]
      },
      "node_12_1537865922924": {
        "0": [
          0
        ]
      },
      "node_22_1537788349430": {
        "1": [
          0,
          0,
          1
        ]
      },
```

```
      "node_27_1537789013960": {
        "1": [
          0
        ]
      },
      "node_20_1537867199210": {
        "1": [
          0,
          0,
          1
        ]
      },
      "node_3_1541521658106": {
        "1": [
          0
        ]
      }
    },
    "dialog_turn_counter": 7,
    "dialog_stack": [
      {
        "dialog_node": "node_3_1541521658106"
      }
    ],
    "dialog_request_counter": 7,
    "initialized": true
  },
  "aceptado": "afirmativo:afirmativo",
  "edad": "25",
  "email": "junior@gmail.com",
  "telefono": "659874895",
  "genero": "Masculino",
  "nombre": "Junior",
  "ocupacion": "Estudiante"
 }
}
```


B Manual del programador

La URL al repositorio donde se encuentra el código es:
https://github.com/MrCxracxn/TFG

C Manual de instalación

Para la instación correcta de esta aplicación solo es necesario instalar los paquetes de Python que vienen indicados en el fichero requeriments.txt del proyecto.